고급기계학습

# 다양한 딥러닝 학습 기법을 이용한 이미지 분류기법

인하대학교 산업경영공학과

12170666 홍상호

12182867 공준식

12180607 김수형

12190625 배기웅

# I. Supervised Learning (지도학습)

## 1.1 모델 설명

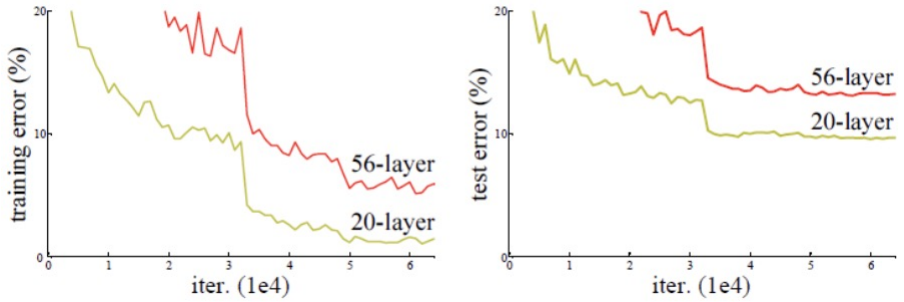
- ResNet
- DRN
- Efficient NET

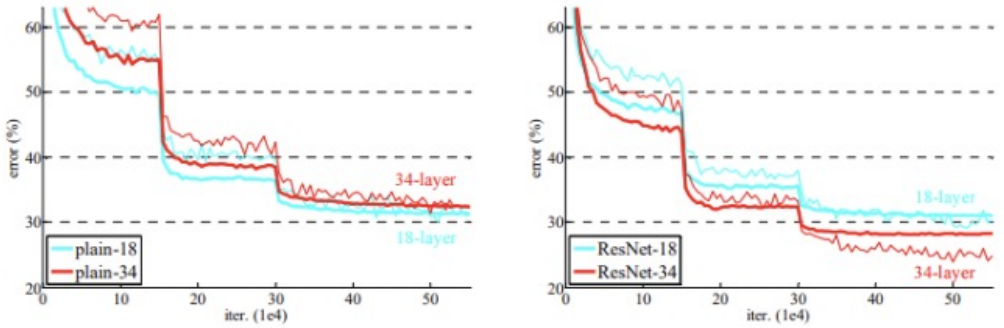
## 1.2 실험 및 실험결과 설명

# 1.1 모델 설명

## ResNet(Residual Network)[1]

- 인간의 시신경 구조가 비전 정보를 처리하는 것을 모방한 모형을 가진 딥러닝 기술인 CNN의  한 종류
- 입력 이미지로부터 정보를 손쉽게 분류하기 위해 저차원의 정보를 고차원으로 확장해서 분류



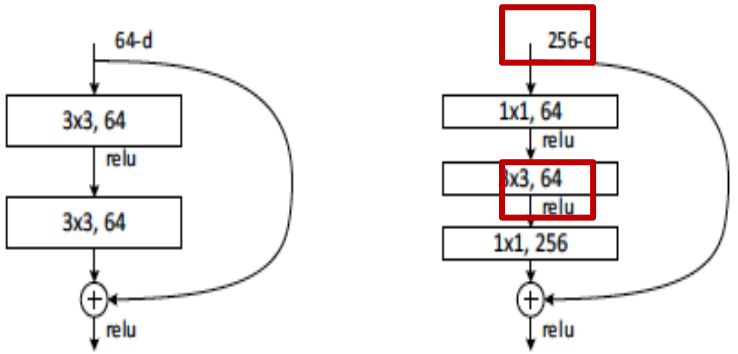

**ACC를 위해 layer을 늘릴수록**
**오히려 ERROR가 높아지는 현상**

**Vanishing gradient**

Layer을 지나간다는 것 = 행렬의 곱

Layer가 깊다는 것 ->  행렬의 곱이 많아짐

**데이터 및 기울기 소실 문제** 발생




**Skip connection**

- **RESNET 18/34**
  - **Basic Block**
- **RESNET 50/101**
  - **BottleNeck**

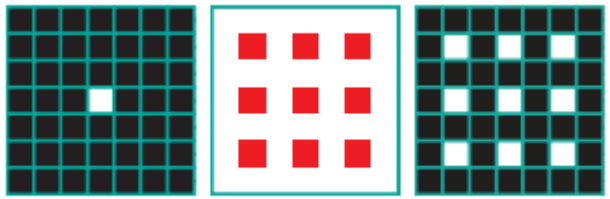
**Ex) input이 256일시에, Dimension을 64로 축소한 후 연산하며, 다시 복원**

# 1.1 모델 설명

## DRN (Dilated Residual Network)[2]

- 이미지 인식과 세분화(segmentation)과 같은 컴퓨터 비전 분야에서 사용되는 딥러닝 모델 구조
- 더욱 정확한 이미지 분류 및 세분화를 위해 ResNet(Residual Network) 구조에 Dilated Convolution을 추가한 것

**Dilation?**




Receptive field의 크기 증가

->필터의 적은 파라미터로 ,**다양한 scale의 이미지에** 대한 대응가능

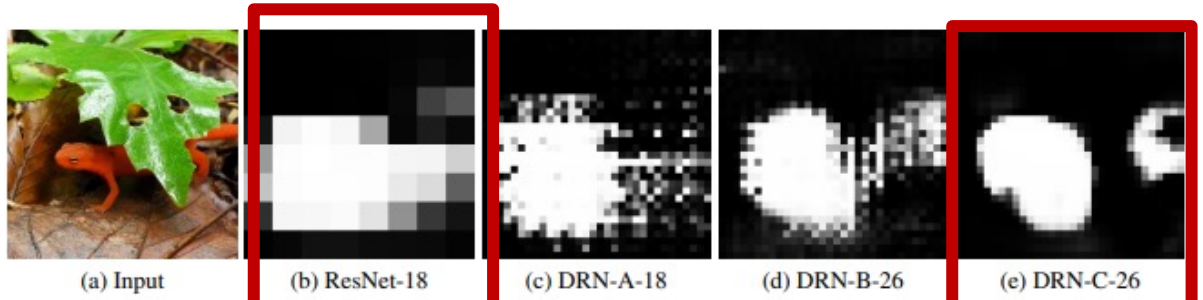



Figure 3: Activation maps of ResNet-18 and corresponding DRNs. A DRN constructed from ResNet-18 as described in Section 2 is referred to as DRN-A-18. The corresponding DRN produced by the degridding scheme described in Section 4 is referred to as DRN-C-26. The DRN-B-26 is an intermediate construction.

→ **고 해상도/ Large scale image에서 높은 Accuracy 를 보임**

- 7X7 영역 연산시,

1-Diliated conv(일반적 CONV) – 49개의 파라미터

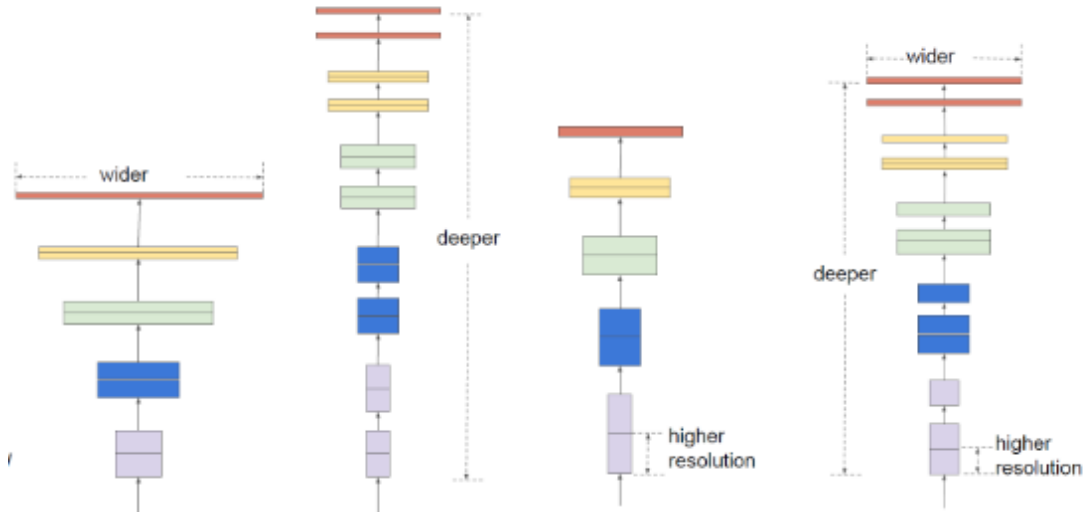
2-Dilated conv – 9개의 파라미터

→ 3X3의 영역의 연산량이 같은 효과

# 1.1 모델 설명

## EFFICIENT NET[3]

- 한정된 자원으로 최대의 효율을 내기 위한 방법으로 model scaling (depth, width, resolution)을 동시에 고려한 것
- 현재 모델 기준으로 **224*224** 해상도의 CIFAR10 DATA에 대해서 **ACC 98%이상** 달성



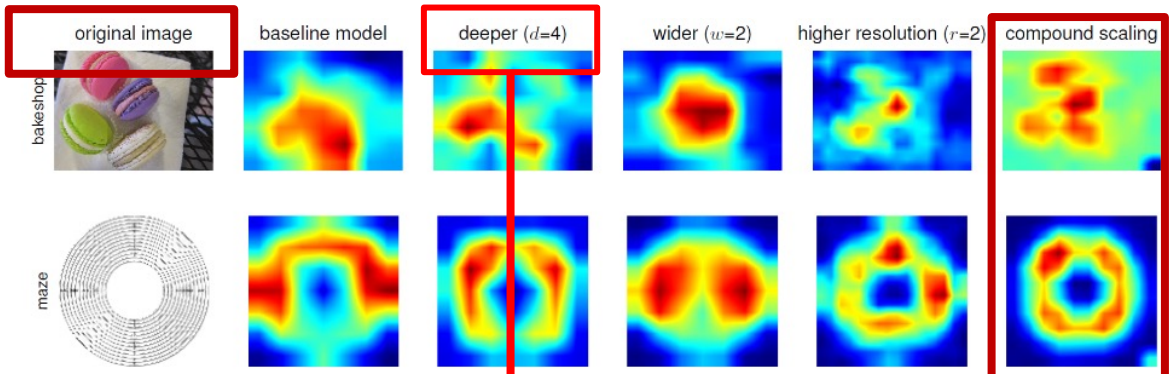




*Figure 7.* **Class Activation Map (CAM) (Zhou et al., 201[illegible]) for Models with different scaling methods-** Our compound scaling method allows the scaled model (last column) to focus on more relevant regions with more object details. Model details are in Table 7.

- **깊이 – ResNet과 같이 layer의 수**
- **넓이 – layer의 width를 늘리면, acc가 늘어나지만, width의 제곱으로 연산량 증가**
- **해상도 – 높은 해상도(픽셀의 수)는 ACC의 증가 BUT, 연산량 또한 증가**

➡ **물체 형태 인식을 좀더 명확하게**

**또한 적은 연산량/빠른 속도/높은 정확도**

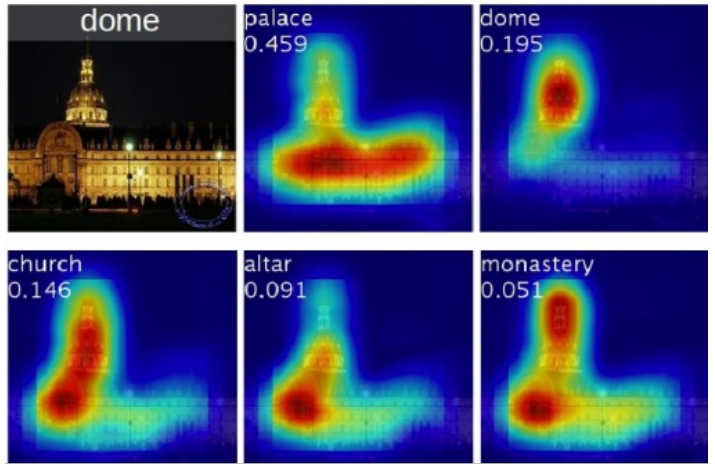
# 1.2 실험 및 실험결과 설명

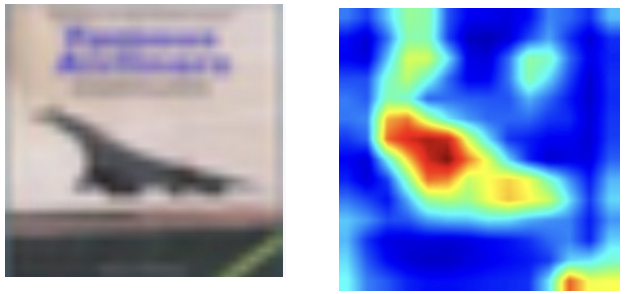
## CAM(Class Activation Map) [4]

CNN의 구조는 **INPUT – CONV LAYERS – FC LAYERS**로 구성되어있는데, FC LAYER의 특성상 FLATTEN을 시킴.

CLASS를 알 수 있지만, CLASS 분류 **근거**를 알 수 없음.

**예시**





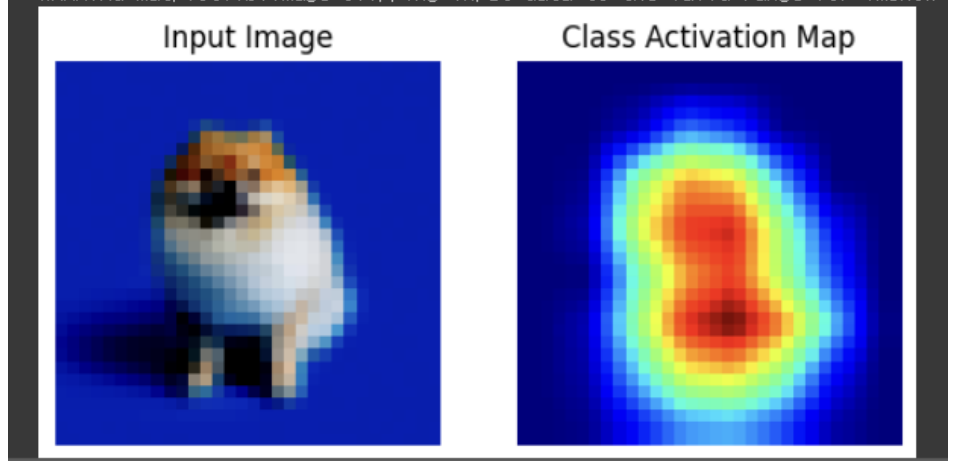
**CLASS 5**(강아지)에 대한 판단 근거의 HEATMAP 생성.

만약, 비행기 CLASS에 대한 HEATMAP을 보여달라 했을 시, 형태를 전혀 잡지 못함.

```
# 모델에 이미지를 입력하여 예측 결과와 CAM을 생성합니다.
output = model(img)
pred = output.argmax(dim=1).item()
cam_img = cam_generator.generate_cam(img, label=5)

# 결과를 시각화합니다.
plt.subplot(121)
plt.imshow(img.squeeze().permute(1, 2, 0).numpy())
plt.title('Input Image')
plt.axis('off')
plt.subplot(122)
plt.imshow(cam_img, cmap='jet')
plt.title('Class Activation Map')
plt.axis('off')
plt.show()
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow
```



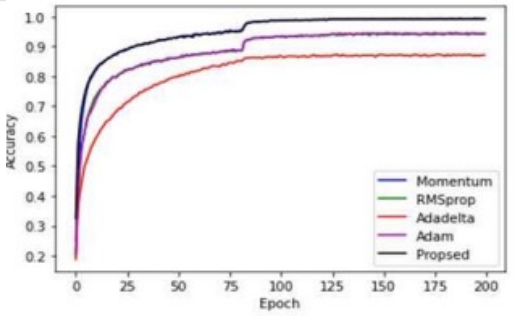
# 1.2 실험 및 실험결과 설명

| MODEL | ACCURACY |
|---|---|
| RESNET 18 | 93.79% |
| RESNET 50 | 94.39% |
| RESNET 101 | 94.5% |
| DRN C 26 | 89.62% |
| DRN D 22 | 89.62% |
| EFFICIENT0 | 86.84% |

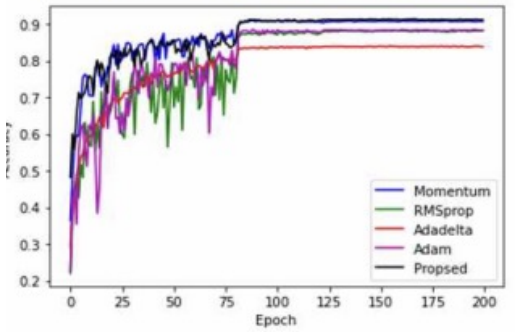
+0.7%

## Accuracy

| Model | Acc. |
|---|---|
| VGG16 | 92.64% |
| ResNet18 | 93.02% |
| ResNet50 | 93.62% |
| ResNet101 | 93.75% |
| RegNetX_200MF | 94.24% |
| RegNetY_400MF | 94.29% |
| MobileNetV2 | 94.43% |
| ResNeXt29(32x4d) | 94.73% |
| ResNeXt29(2x64d) | 94.82% |
| SimpleDLA | 94.89% |
| DenseNet121 | 95.04% |
| PreActResNet18 | 95.11% |
| DPN92 | 95.16% |
| DLA | 95.47% |




(a) Accuracy of training data for each algorithm.




(b) Accuracy of test data for each algorithm.

ig. 6. Performance on the CIFAR−10 dataset

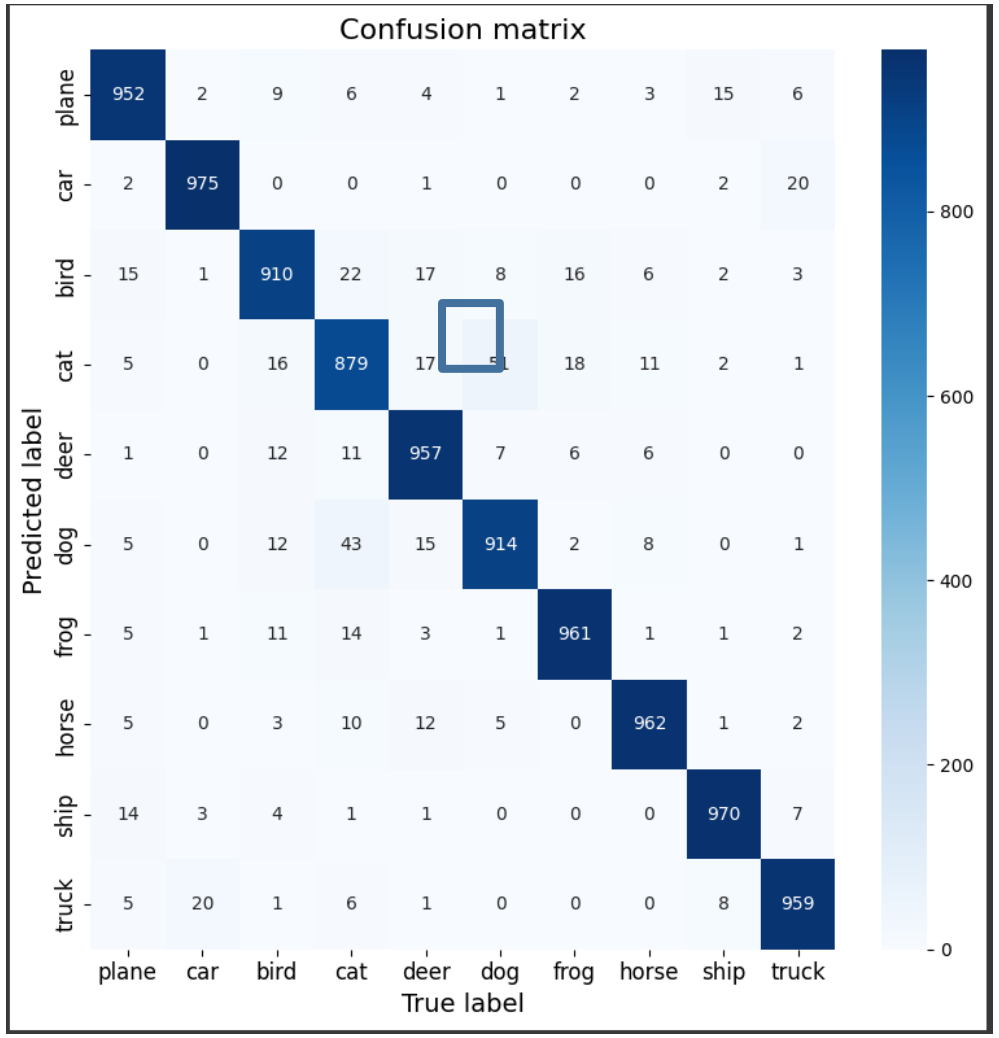
The results of the test AUC of the SGD optimizer.

| Test AUC | | |
|---|---|---|
| Batch size | SGD LR = 0.0001 | SGD LR = 0.001 |
| 16 | 0.9555 | 0.9461 |
| 32 | **0.9570** | 0.9521 |
| 64 | 0.9512 | 0.9545 |
| 128 | 0.9302 | 0.9567 |
| 256 | 0.9077 | **0.9579** |

| 하이퍼파라미터 | Scale |
|---|---|
| 학습률 | 0.001 |
| 배치 사이즈 | 128 |
| 스케쥴러 | StepLR |
| 옵티마이저 | SGD + Momentum |
| Transform | RandomCrop (Padding – 4) RandomHoriziontalFlip Normalize (0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010) |

# 1.2 실험 및 실험결과 설명

## CONFUSION MATRIX




```
[21] print_classification_report(model, test_loader, classes)
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| plane | 0.94 | 0.95 | 0.95 | 1000 |
| car | 0.97 | 0.97 | 0.97 | 1000 |
| bird | 0.93 | 0.91 | 0.92 | 1000 |
| cat | 0.89 | 0.88 | 0.88 | 1000 |
| deer | 0.93 | 0.96 | 0.94 | 1000 |
| dog | 0.93 | 0.91 | 0.92 | 1000 |
| frog | 0.96 | 0.96 | 0.96 | 1000 |
| horse | 0.96 | 0.96 | 0.96 | 1000 |
| ship | 0.97 | 0.97 | 0.97 | 1000 |
| truck | 0.96 | 0.96 | 0.96 | 1000 |
| accuracy | | | 0.94 | 10000 |
| macro avg | 0.94 | 0.94 | 0.94 | 10000 |
| weighted avg | 0.94 | 0.94 | 0.94 | 10000 |

- CAT을 제외한

-> 9개의 CLASS에서는 평균 95%의 ACC

# II. Unsupervised Learning (비지도학습)

2.1 비지도학습 모델 종류

- VAE(Variational Auto Encoder )
- DCGAN(Deep Convolutional Generative Adversarial Network)
- DDPM(Denoising Diffusion Probabilistic Model)

2.2 본 실험 과정에서 비지도학습의 한계

2.3 실험 및 실험 결과

# 2.1 비지도학습 모델 종류

**Deep Belief Network (2006)**

SRGAN **(2017)** Cycle GAN **(2017)** Stable Diffusion **(2022)**

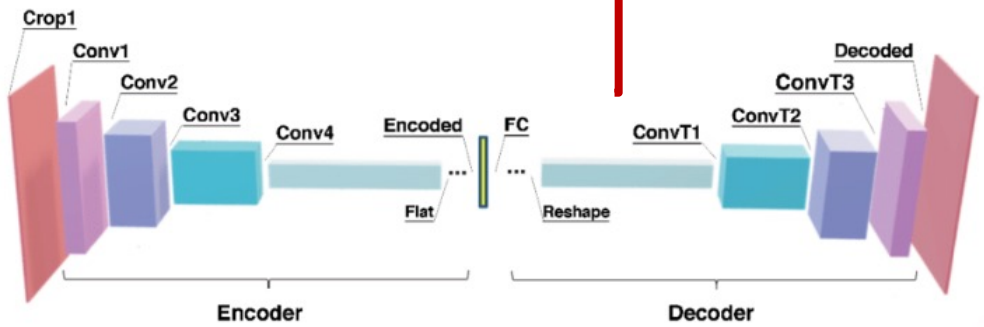
**(1) Variational Autoencoder (2014) [5]**

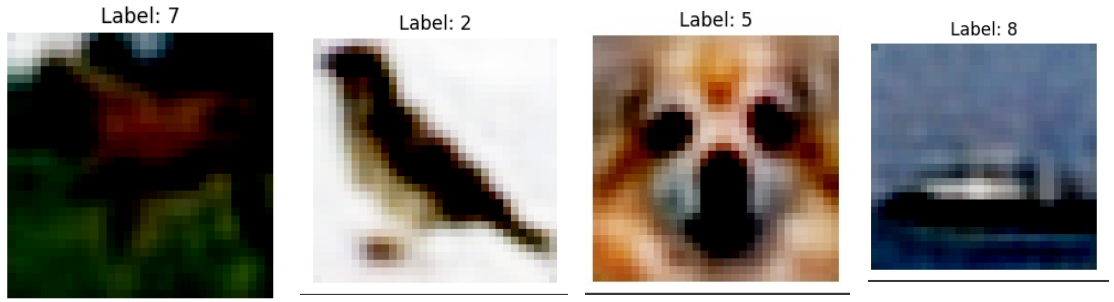
Generative Adversarial Network (2014)

Diffusion Generation Model (2020)

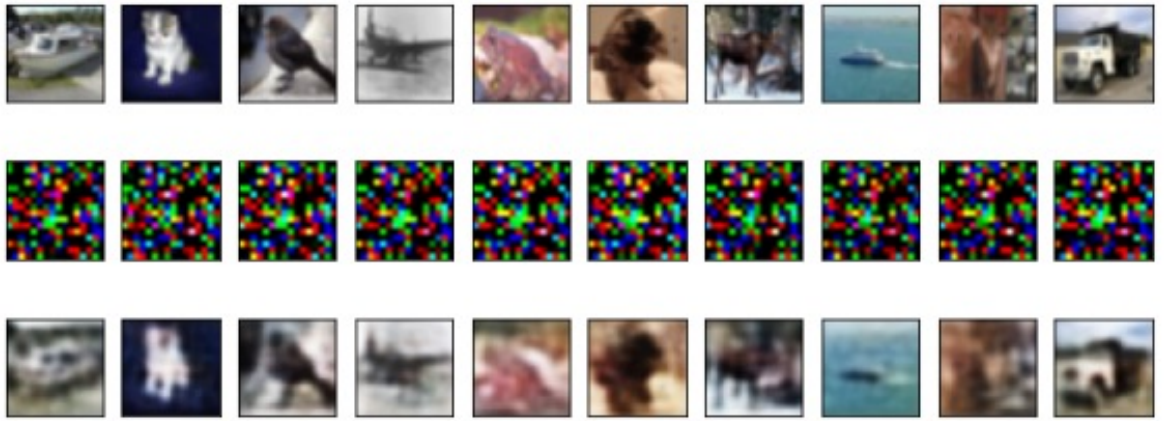
'필연적인 정보 손실'의 발생.
-> 얼마나 '중요한 특징'들을 남겨두는가.

차원의 축소




- Normal / Encoder/ Decoder



**입력 데이터를 최대한 compression 시킨 후, compressed data를 다시 본래의 입력 형태로 복원 시키는 신경망.**

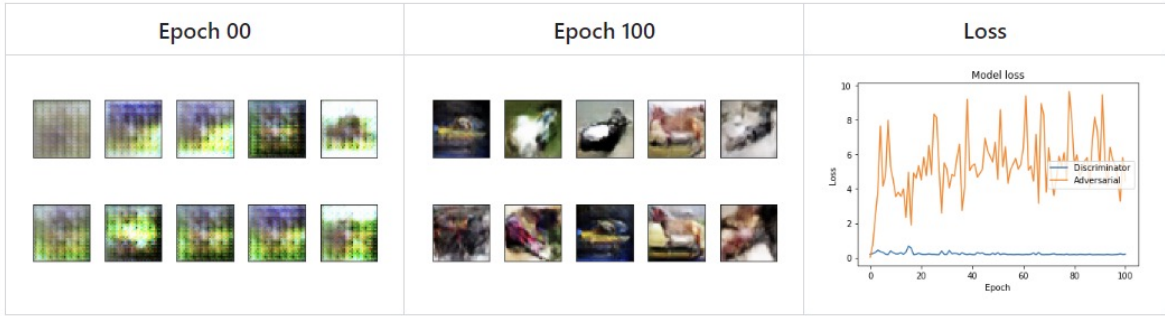
# 2.1 비지도학습 모델 종류

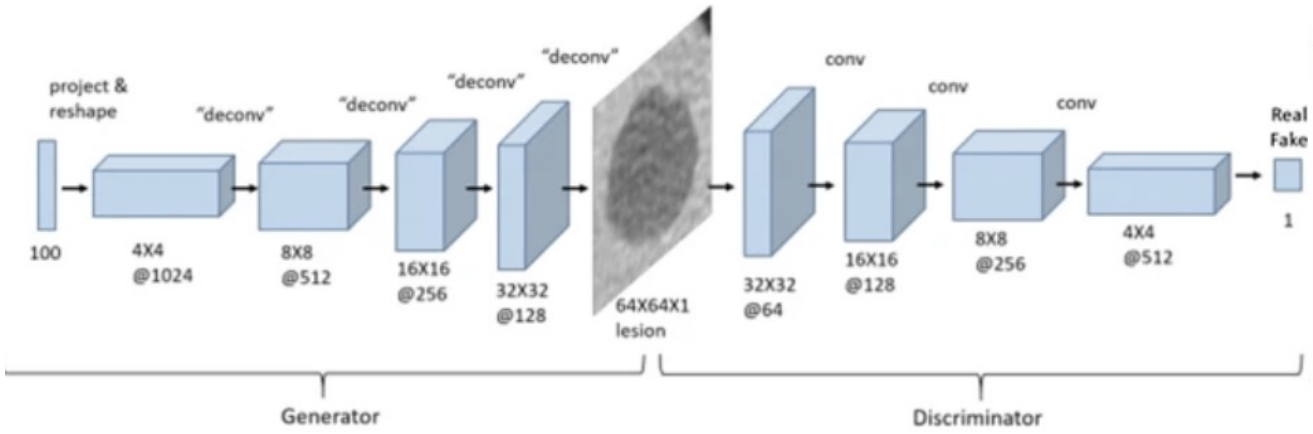
Deep Belief Network (2006)

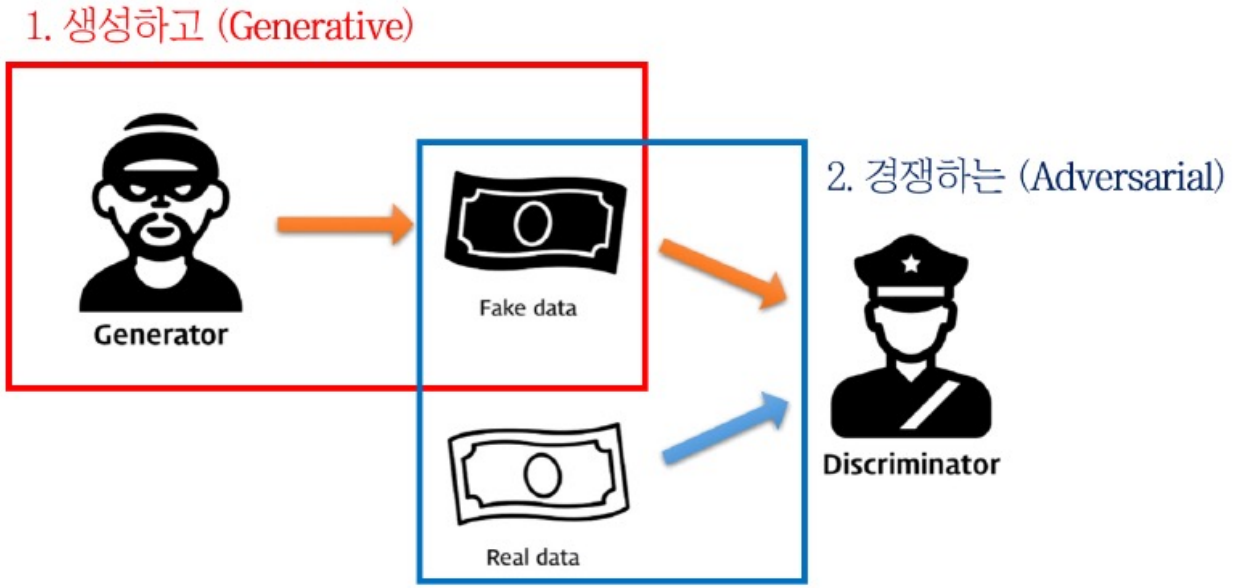
Variational Autoencoder (2014) **(2) Generative Adversarial Network (2014) [6]** DCGAN (2016) SRGAN **(2017)** Cycle GAN **(2017)** Stable Diffusion **(2022)** Diffusion Generation Model (2020

**생성자와 구분자에서 합성곱(convolution) 을 사용**

**-> 특정 filter들이 이미지의 특정 물체를 학습했다는 것을 보여주며, 벡터 산술 연산이 가능하여 이미지 생성이 가능.**



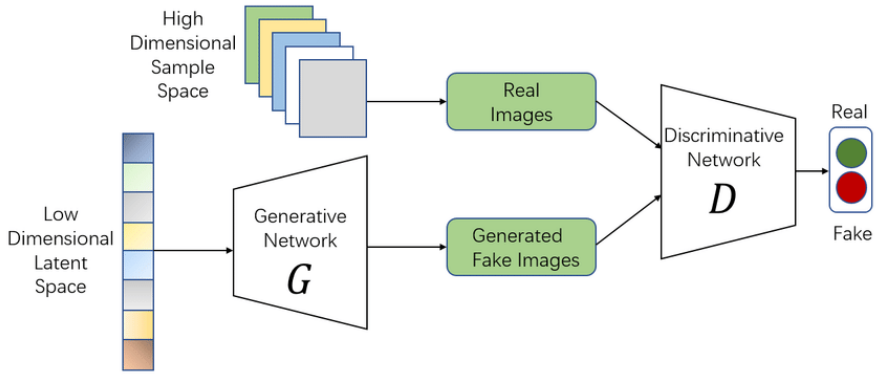
# 2.1 비지도학습 모델 종류

**Deep Belief Network (2006)**

Variational Autoencoder (2014) **(2) Generative Adversarial Network (2014) [6]** **DCGAN (2016)** SRGAN **(2017)** Cycle GAN **(2017)** Stable Diffusion **(2022)** Diffusion Generation Model (2020

**GAN의 학습과정**

1. 임의의 noise로 구성된 latent vector로부터 생성자(**G**)는 이미지를 생성
2. 실제 이미지는 진짜(1)로, 생성된 이미지는 가짜(0)로 판별자는 학습
3. **G**는 **D**의 피드백을 반영하여 **D**가 진짜라고 판별할 수 있도록 그럴싸한 이미지를 생성하기 위해 학습
4. 위와 같은 과정을 계속 반복하여 **D**가 생성된 이미지를 진짜 이미지로 판별하게 되면 학습 완료



# 2.1 비지도학습 모델 종류

**Deep Belief Network (2006)**

Variational Autoencoder (2014) **(2) Generative Adversarial Network (2014) [6]** **DCGAN (2016)** SRGAN **(2017)** Cycle GAN **(2017)** Stable Diffusion **(2022)** Diffusion Generation Model (2020

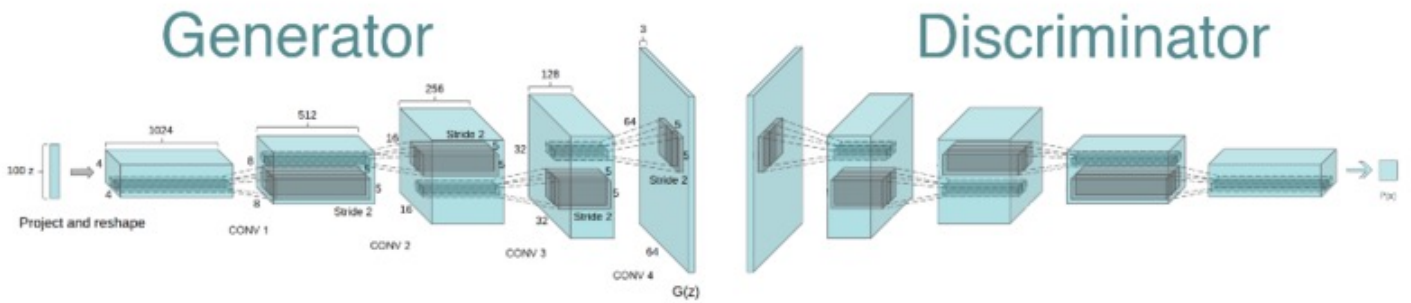
**DCGAN의 특징**

1. 기존 GAN의 완전연결층과 풀링층을 최대한 배제하고, strided convolution과 transposed convolution으로 네트워크가 구성됨. 이를 통해 이미지의 위치 정보를 파악할 수  있음.

2. 배치 정규화(Batch Normalization)을 사용하였음. 이는 입력 데이터가 치우쳐져 있을 경우의 평균과 분산을 조정해주는 역할을 함. 따라서 역전파를 시행하였을 때 각 층에 가중치가 전달이 잘 될 수 있도록 안정적으로 학습을 할 수 있음

3. 모든 층에 활성화함수로 ReLU함수를 사용함. 마지막에는  tanh함수를 사용.

4. 판별자의 모든 층에는 LeakyReLU함수를 사용하였음.

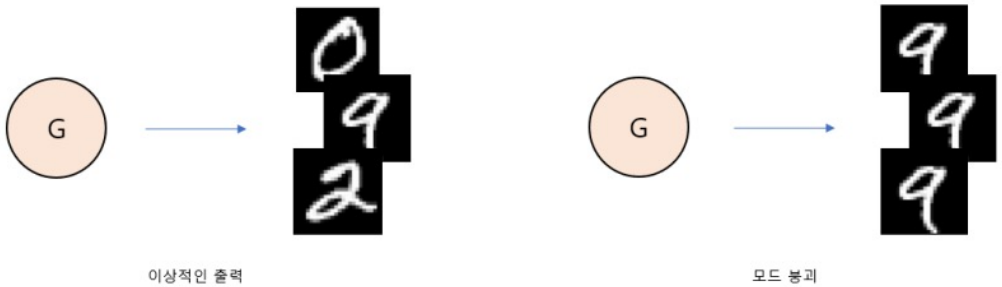


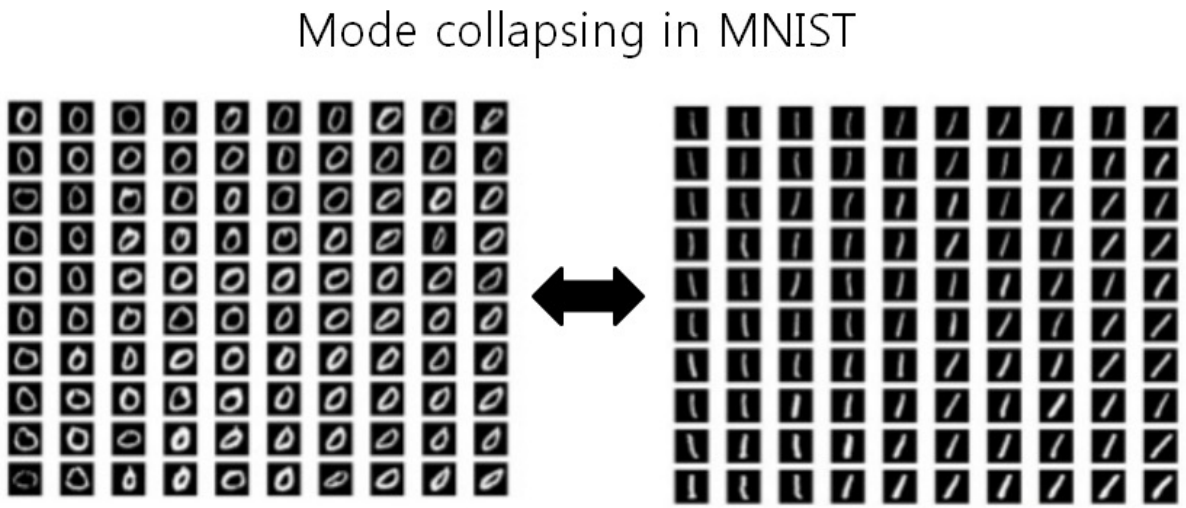
# 2.1 비지도학습 모델 종류

**Deep Belief Network (2006)**

DCGAN (2016) SRGAN **(2017)** Cycle GAN **(2017)** Stable Diffusion **(2022)**

Variational Autoencoder (2014) **(2) Generative Adversarial Network (2014) [6]** Diffusion Generation Model (2020




**GAN계열 모델의 단점**

- 생성자 모델과 판별자 모델이 같이 학습이 되어야 하며 그 과정에서 모드 붕괴현상이 발생할 수 있음
- 생성자 모델은 어떤 이미지를 생성하든 판별자 모델만 속이면 되기 때문에 판별자 모델이 불완전한 경우, 학습 데이터의 일부분만 학습하게 됨
- 이렇게 학습 데이터 전체의 분포를 학습하지 못하면, 생성자 모델이 표본을 다양하게 생성하지 못하게 될 수 있음

# 2.1 비지도학습 모델 종류

**Deep Belief Network (2006)**

DCGAN (2016) SRGAN **(2017)** Cycle GAN **(2017)** Stable Diffusion **(2022)**

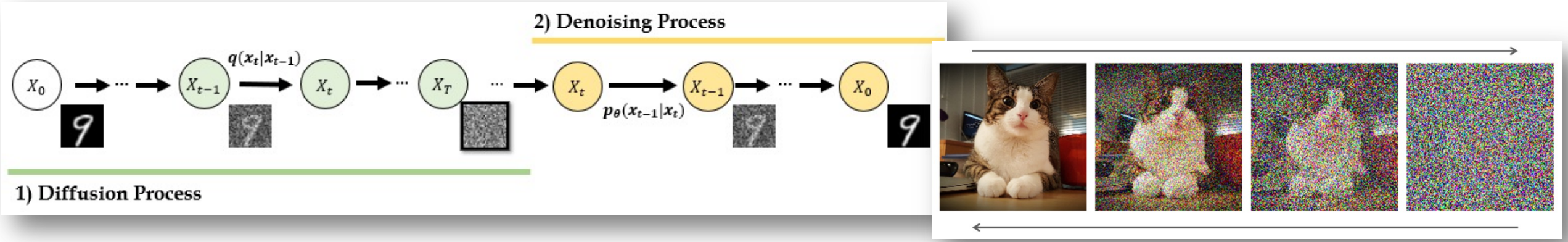
Variational Autoencoder (2014) Generative Adversarial Network (2014) **(3) Diffusion Generation Model (2020) [9]**

- 디퓨전 생성모델은 생성적 적대 신경망의 이러한 문제점을 해결하기 위해 등장하였음.

- 디퓨전 생성모델의 학습과정은 크게 (1) Diffusion Process와 (2)Denoising Process로 나뉨.

학습과정은 마코프체인을 기반으로 하며, $t-1$번째 학습이 이루어진 후 t번째로 넘어가는 구조임.

- Denoising Process를 통해 노이즈를 걷어내어 생성하는 과정을 학습하여 이미지를 생성하는 모델.



# 2.1 비지도학습 모델 종류

| | VAE [5] | DCGAN [6,7] | DDPM [9] |
|---|---|---|---|
| 특징 | 1. 인코더와 디코더를 활용해 잠재공간을 도출<br>2. 이 잠재공간으로부터 원하는 출력 이미지를 디코딩함으로써 데이터를 생성 | 데이터를 생성하는 생성자모델과 데이터를 구별하는 판별자모델이 경쟁하는 과정을 통해서 데이터를 학습 및 생성 | 원본이미지에 가우시안 노이즈를 조금씩 더해가면서 noise로 만드는 diffusion process와 , 임의의 노이즈로부터 조금씩 복원해가는 denoising process를 통해 데이터를 학습 및 생성 |
| 장점 | 레이블을 고정한 상태로 생성할 수 있다.<br>이미지 생성속도가 빠르다. | 이미지를 비교적 빠르고 정밀하게 생성할 수 있다. | 높은 성능의 이미지를 정교하게 생성할 수 있음.<br>다양한 이미지를 생성할 수 있으며,<br>모드붕괴현상을 해결할 수 있음. |
| 단점 | 다른 모델에 비해 정교하고 높은 성능의 이미지를 생성하기 힘들다. | GAN 계열 모델인 DCGAN은 판별자가 판별하기 어려운 특징의 이미지만 생성하는 모드붕괴현상 [8]이 발생한다.<br>판별자와 생성자 두 모델이 학습해야 한다. | 생성하는데 시간이 많이 걸린다. |

# 2.2 본 실험 과정에서 비지도학습의 한계

1. 지도학습을 위해서는 이미지 뿐만 아니라 이미지에 해당하는 레이블링도 필요함.

# 2.2 본 실험 과정에서 비지도학습의 한계

1. 지도학습을 위해서는 이미지 뿐만 아니라 이미지에 해당하는 레이블링도 필요함.

2. VAE는 처음부터 레이블을 가진 원본을 그대로 가져와 축소 및 확장을 하므로 레이블 보존이 비교적 쉬움. -> 실험결과 6-70%의 accuracy로 이미지 보강의 의미가 없었음.

# 2.2 본 실험 과정에서 비지도학습의 한계

1. 지도학습을 위해서는 이미지 뿐만 아니라 이미지에 해당하는 레이블링도 필요함.

2. VAE는 처음부터 레이블을 가진 원본을 그대로 가져와 축소 및 확장을 하므로 레이블 보존이 비교적 쉬움. -> 실험결과 6-70%의 accuracy로 이미지 보강의 의미가 없었음.

3. 반면 DCGAN과 DDPM은 랜덤 생성이므로, 레이블링이 어렵다는 한계가 있음.

# 2.2 본 실험 과정에서 비지도학습의 한계

1. 지도학습을 위해서는 이미지 뿐만 아니라 이미지에 해당하는 레이블링도 필요함.

2. VAE는 처음부터 레이블을 가진 원본을 그대로 가져와 축소 및 확장을 하므로 레이블 보존이 비교적 쉬움. -> 실험결과 6-70%의 accuracy로 이미지 보강의 의미가 없었음.

3. 반면 DCGAN과 DDPM은 랜덤 생성이므로, 레이블링이 어렵다는 한계가 있음.

4. 따라서 DCGAN과 DDPM은 "이미지 보강"이 아닌, **"이미지 보강에 적합한 고품질 이미지 생성 여부"**에 중점을 두어 실험을 진행하였음.

# 2.2 본 실험 과정에서 비지도학습의 한계

1. 지도학습을 위해서는 이미지 뿐만 아니라 이미지에 해당하는 레이블링도 필요함.

2. VAE는 처음부터 레이블을 가진 원본을 그대로 가져와 축소 및 확장을 하므로 레이블 보존이 비교적 쉬움. -> 실험결과 6-70%의 accuracy로 이미지 보강의 의미가 없었음.

3. 반면 DCGAN과 DDPM은 랜덤 생성이므로, 레이블링이 어렵다는 한계가 있음.

4. 따라서 DCGAN과 DDPM은 "이미지 보강"이 아닌, **"이미지 보강에 적합한 고품질 이미지 생성 여부"**에 중점을 두어 실험을 진행하였음.

5. DCGAN과 DDPM으로 이미지를 생성하고, 평가지표인 FID Score를 이용하여 성능을 비교하였음.

# 2.3 실험 및 실험 결과

## FID (Fréchet Inception Distance) [10]

- 생성된 이미지의 분포와 원래 이미지의 분포가 어느정도 비슷한 지 측정하는 지표
- ImageNet으로 사전 훈련된 Inception Model을 FID 평가를 위한 모델로 사용하고, Inception Model 의 마지막 특징 맵인 1024차원 벡터를 FID 측정에 사용
- 5000장의 이미지를 생성하여 FID Score 계산에 사용
- $d^2((m, C), (m_w, C_w)) = \|m - m_w\|_2^2 + Tr(C + C_w - 2(CC_w)^{1/2})$

Inception Model

Feature map 1

$(\mu, C)$

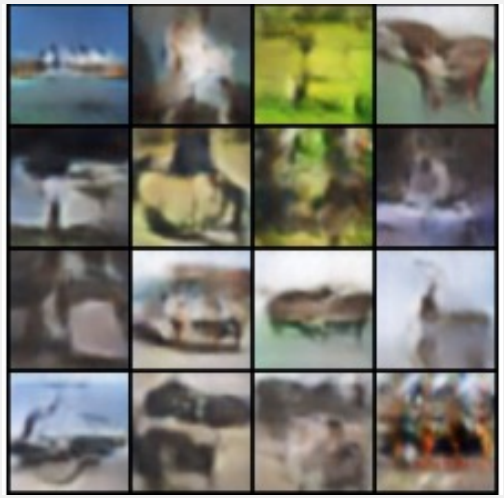
Inception Model

Feature map 2

$(\mu_w, C_w)$

# 2.3 실험 및 실험 결과

## 실험 결과

| 모델 명 | DCGAN | DDPM |
|---|---|---|
| FID Score | 189.13 | 35.33 |
| (lr, batch size, epoch) | $(1 * 10^{-4}, 128, 200)$ | $(2 * 10^{-4}, 128, 50)$ |
| 실험비교 결과 | - DDPM이 더 다양한 이미지를 학습<br>- DDPM이 생성한 이미지가 더 깨끗하고 선명함.<br>- 생성 시간은 DCGAN이 더 빨랐음. | |
| 생성한 이미지 |  |  |

# III. Semi Supervised Learning(준지도학습)

3.1 준지도학습이란?

3.2 실험 및 실험 결과

- Hyper-parameter Table
- Ideal Results & Attemps
- Visualize results using Tensorboards

# 3.1 준지도학습이란?

## Semi Supervised Learning (SSL)

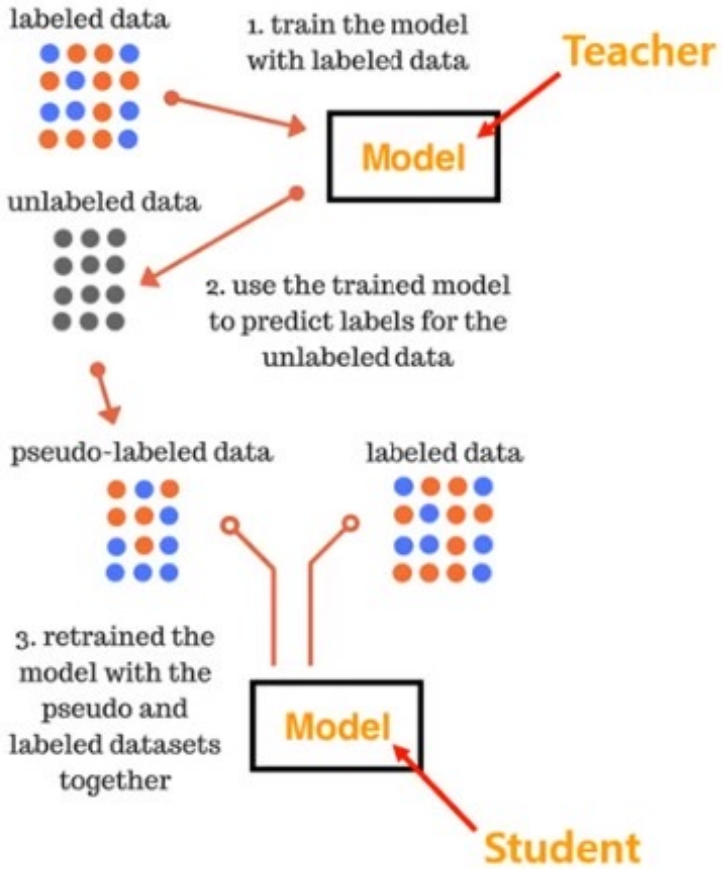
- 기존 Supervised learning은 labeling 작업이 필요

  → 이러한 과정에는 상당한 노력과 비용이 듦

- 반면 SSL은

  정답 레이블이 있는 작은 데이터셋으로 1차 (지도)학습 후
  정답 레이블이 없는 큰 데이터셋으로 2차 학습

  → labeling 작업을 줄일 수 있어 활발히 연구되고 있음



# 3.1 준지도학습이란?

## Meta pseudo Labels [11]

| | Method | CIFAR-10-4K (mean ± std) | SVHN-1K (mean ± std) | ImageNet-10% Top-1 | ImageNet-10% Top-5 |
|---|---|---|---|---|---|
| Label Propagation Methods | Temporal Ensemble [35] | 83.63 ± 0.63 | 92.81 ± 0.27 | – | |
| | Mean Teacher [64] | 84.13 ± 0.28 | 94.35 ± 0.47 | – | |
| | VAT + EntMin [44] | 86.87 ± 0.39 | 94.65 ± 0.19 | – | 83.39 |
| | LGA + VAT [30] | 87.94 ± 0.19 | 93.42 ± 0.36 | – | |
| | ICT [71] | 92.71 ± 0.02 | 96.11 ± 0.04 | – | |
| | MixMatch [5] | 93.76 ± 0.06 | 96.73 ± 0.31 | – | |
| | ReMixMatch [4] | 94.86 ± 0.04 | 97.17 ± 0.30 | – | |
| | EnAET [72] | 94.65 | 97.08 | – | |
| | FixMatch [58] | 95.74 ± 0.05 | 97.72 ± 0.38 | 71.5 | 89.1 |
| | UDA* [76] | 94.53 ± 0.18 | 97.11 ± 0.17 | 68.07 | 88.19 |
| Self-Supervised Methods | SimCLR [8, 9] | – | – | 71.7 | 90.4 |
| | MOCOv2 [10] | – | – | 71.1 | – |
| | PCL [38] | – | – | – | 85.6 |
| | PIRL [43] | – | – | – | 84.9 |
| | BYOL [21] | – | – | 68.8 | 89.0 |
| | **Meta Pseudo Labels** | **96.11 ± 0.07** | **98.01 ± 0.07** | **73.89** | **91.38** |
| | Supervised Learning with full dataset* | 94.92 ± 0.17 | 97.41 ± 0.16 | 76.89 | 93.27 |

Interestingly, on CIFAR-10-4K, Meta Pseudo Labels even exceeds the headroom supervised learning on full dataset.

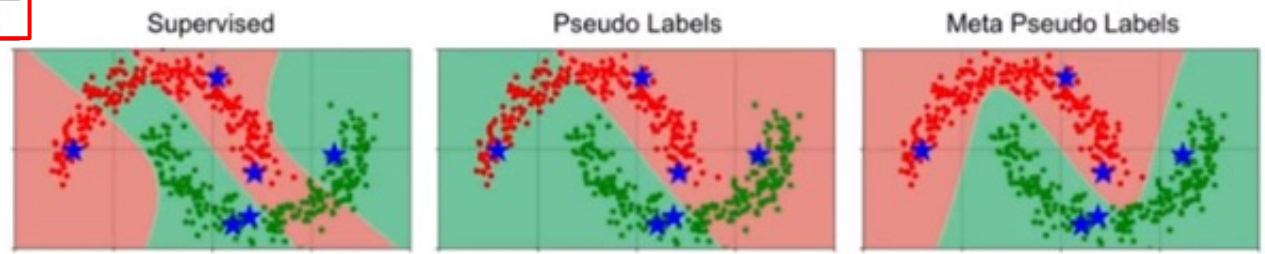



Figure 3. Meta Pseudo Label performance on TwoMoon dataset [1]

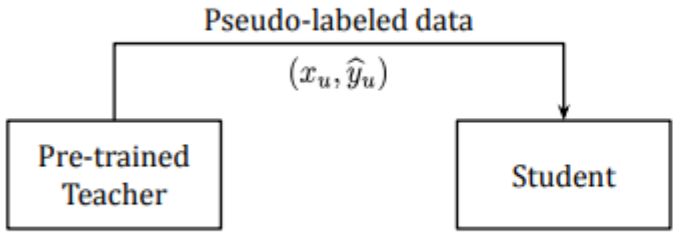
Creates a more accurate decision boundary than other models in TwoMoon dataset

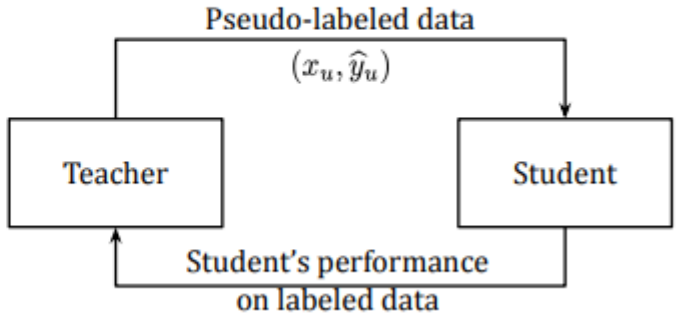
# 3.1 준지도학습이란?

**Meta Pseudo Label's Key Algorithms**

기존 pseudo label 을 한단계 업그레이드 시킨 모델

Student – Teacher Feedback 을 통해 더 정확한 pseudo label을 생성 가능




Create pseudo-labeled data by pre-trained Teacher model that trained Labeled data.
Then learn the student model with psedo_labeled data

- Teacher 성능을 student 가 뛰어넘을 수 없음.
- Teacher 모델의 성능에 따라 pseudo label에 대한 퀄리티 저하와 Student 학습 성능 저하 발생가능

By learning the Teacher model together , continue to update the Teacher model with the Student model's learning performance feedback

- Feed-back을 통한 meta learning 과정으로 Teacher 성능을 향상시킴

# 3.1 준지도학습이란?

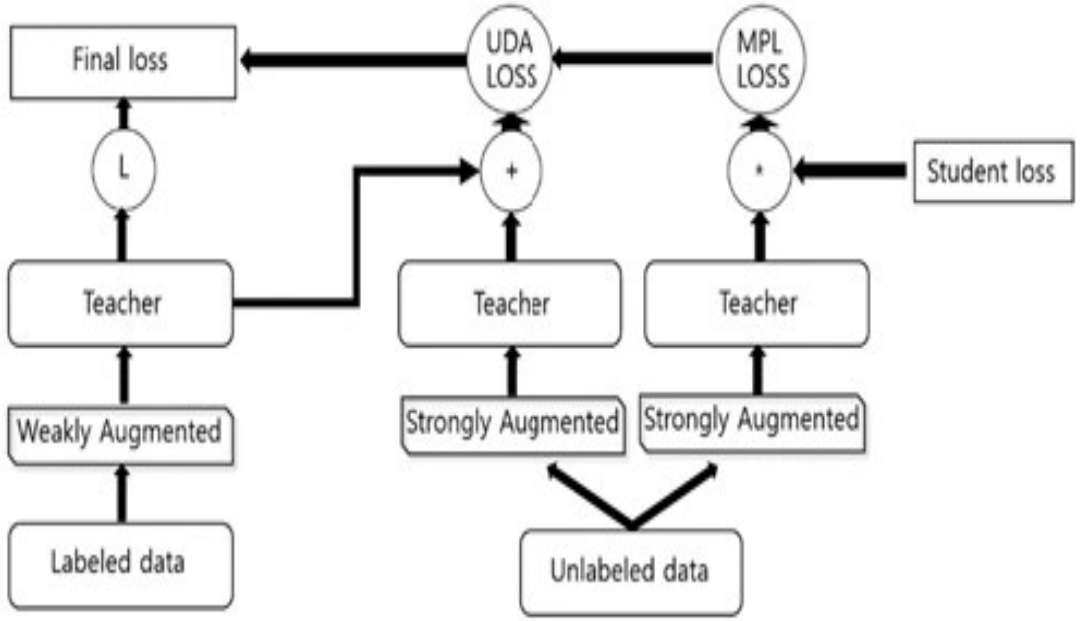
## Introduction to key techniques applied to the model

- **Number of Label and unlabel Data**

  : Of the 50,000 images, 5,000 images are for validation,
  4,000 imgaes are used as labeled data
  41,000 images are used as unlabeled data

- **Data augmentation**

  : RandAugment for UDA
  (Apply augmentation to pseudo-labeld data)




- **Algorithm for Meta Pseudo Labels with UDA**

  : Sample an unlabeled example $X_u$ and a labeled example $X_i$, $Y_i$

  Sample a pseudo label

  0. Update the student using the pseudo label
  1. Compute the teacher's gradient from the student's feedback
  2. Compute the teacher's gradient on labeled data
  3. Compute the teacher's gradient on the UDA loss with unlabeled data

  Update the teacher

# 3.2 실험 및 실험결과

## Hyper-parameter Table

**Optimizer** : SGD (Stochastic Gradient Descent)

**Model** : Wide ResNet

| | Hyper-parameter | CIFAR-10 |
|---|---|---|
| **Commons** | Weight decay | 0.0005 |
| | Label smoothing | 0.15 |
| | Number of traing steps | 30,000 |
| | Number of eval-step | 1,000 |
| **Student** | Learning rate | 0.05 |
| | Batch size | 128 |
| | Dropout rate | 0.2 |
| **Teacher** | Learning rate | 0.05 |
| | Batch size | 128 |
| | Dropout rate | 0.2 |
| | UDA step | 5000 |
| **Finetune** | Epochs | 625 |
| | Batch size | 512 |
| | Learning rate | 0.00003 |
| | Momentum | 0.9 |

# 3.2 실험 및 실험결과

## Ideal result

### Results

| | CIFAR-10-4K |
|---|---|
| Paper (w/ finetune) | 96.11 ± 0.07 |
| This code (w/o finetune) | 96.01 |
| This code (w/ finetune) | 96.08 |

```
--total-steps 300000
```

# 3.2 실험 및 실험결과

## Attempts

```
05/14/2023 14:55:01 - INFO - __main__ -   top-1 acc: 95.23
05/14/2023 14:55:01 - INFO - __main__ -   Best top-1 acc: 95.36
Train Iter: 129000/300000. LR: 0.0316. Data: 0.03s. Batch: 0.25s. S_Loss: 0.8996. T_Loss: 4.1259. Mask: 0.9833. : 100% 1000/1000 [04:12<00:00,  3.97it/s]
Test Iter:  79/ 79. Data: 0.01s. Batch: 0.01s. Loss: 0.7937. top1: 95.03. top5: 99.96. : 100% 79/79 [00:01<00:00, 72.20it/s]
05/14/2023 14:59:14 - INFO - __main__ -   top-1 acc: 95.03
05/14/2023 14:59:14 - INFO - __main__ -   Best top-1 acc: 95.36
Train Iter: 129656/300000. LR: 0.0315. Data: 0.03s. Batch: 0.25s. S_Loss: 0.8996. T_Loss: 4.1183. Mask: 0.9836. :  66% 656/1000 [02:46<01:18,  4.37it/s]
```

**1. Total step : 300000 / Top-1 acc :95.03 / Best top-1 acc : 95.36 w/o finetune**

```
05/15/2023 13:07:14 - INFO - __main__ -   top-1 acc: 94.19
05/15/2023 13:07:14 - INFO - __main__ -   Best top-1 acc: 94.21
Train Iter: 39000/100000. LR: 0.0373. Data: 0.04s. Batch: 0.78s. S_Loss: 0.9456. T_Loss: 4.4126. Mask: 0.9759. : 100% 1000/1000 [13:00<00:00,  1.28it/s]
Test Iter:  79/ 79. Data: 0.01s. Batch: 0.03s. Loss: 0.8096. top1: 94.31. top5: 99.88. : 100% 79/79 [00:02<00:00, 32.09it/s]
05/15/2023 13:20:17 - INFO - __main__ -   top-1 acc: 94.31
05/15/2023 13:20:17 - INFO - __main__ -   Best top-1 acc: 94.31
Train Iter: 39982/100000. LR: 0.0365. Data: 0.03s. Batch: 0.77s. S_Loss: 0.9421. T_Loss: 4.3946. Mask: 0.9765. :  98% 982/1000 [12:42<00:13,  1.35it/s]
```

8시간 40분 29초 오후 10:33에 완료됨

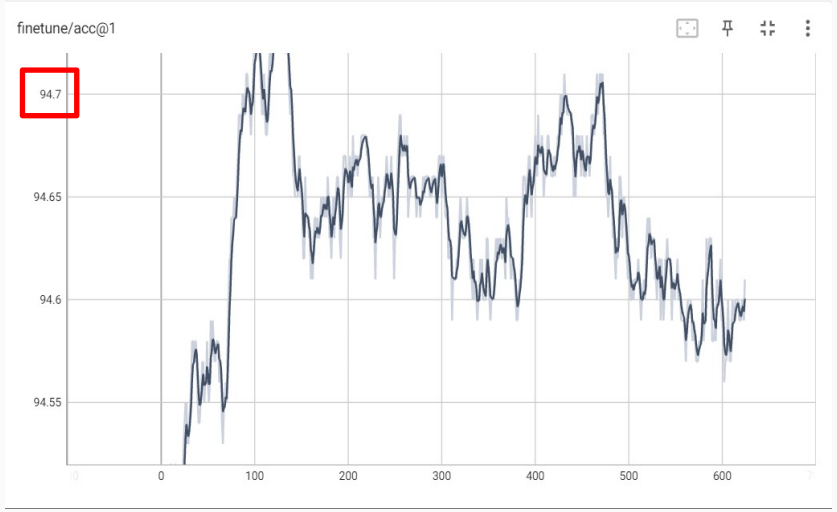
**2. Total step : 100000 / Top-1 acc :94.31 / Best top-1 acc : 94.31 w/o finetune**

```
05/16/2023 07:45:46 - INFO - __main__ -   top-1 acc: 94.49
05/16/2023 07:45:46 - INFO - __main__ -   Best top-1 acc: 94.49
Train Iter: 30000/30000. LR: 0.0000. Data: 0.04s. Batch: 0.80s. S_Loss: 0.8454. T_Loss: 3.5204. Mask: 0.9923. : 100% 1000/1000 [13:16<00:00,  1.25it/s]
Test Iter:  79/ 79. Data: 0.01s. Batch: 0.02s. Loss: 0.8125. top1: 94.49. top5: 99.78. : 100% 79/79 [00:01<00:00, 48.31it/s]
```

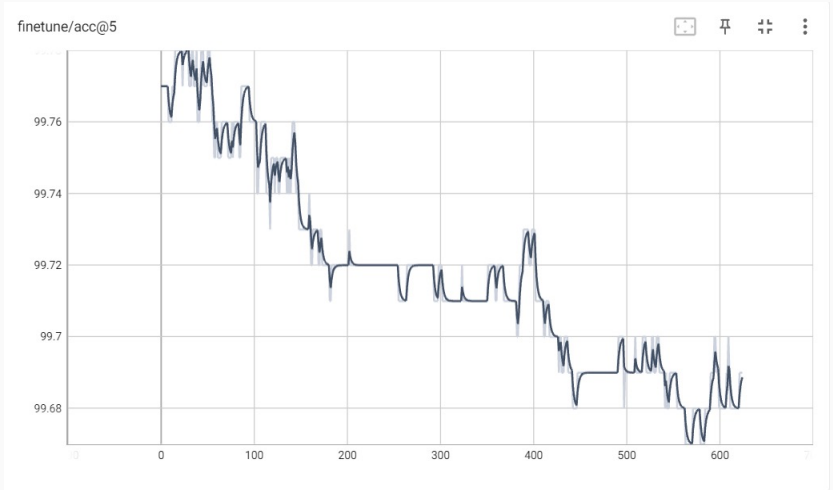
```
Finetune Epoch: 625/625. Data: 1.34s. Batch: 1.48s. Loss: 0.8183. : 100% 8/8 [00:01<00:00,  4.13it/s]
Test Iter:  79/ 79. Data: 0.01s. Batch: 0.02s. Loss: 0.8133. top1: 94.61. top5: 99.69. : 100% 79/79 [00:01<00:00, 49.61it/s]
05/16/2023 08:36:28 - INFO - __main__ -   top-1 acc: 94.61
05/16/2023 08:36:28 - INFO - __main__ -   Best top-1 acc: 94.73
```

# 3.2 실험 및 실험결과

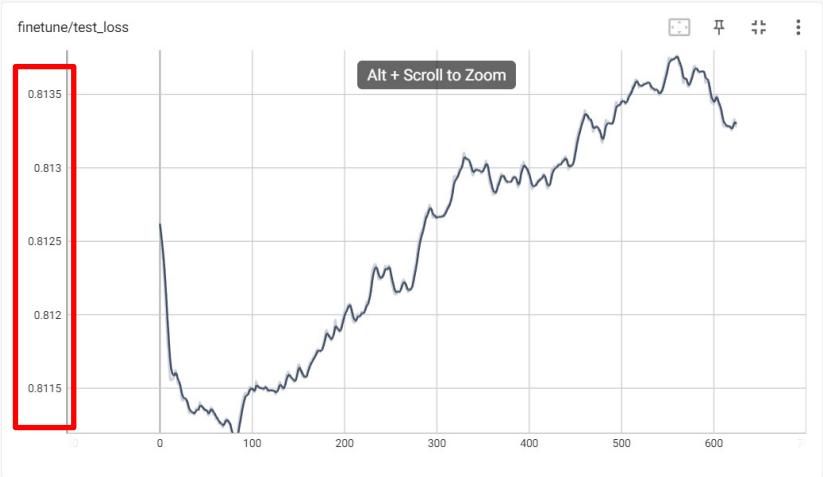
Top-1 Accuracy



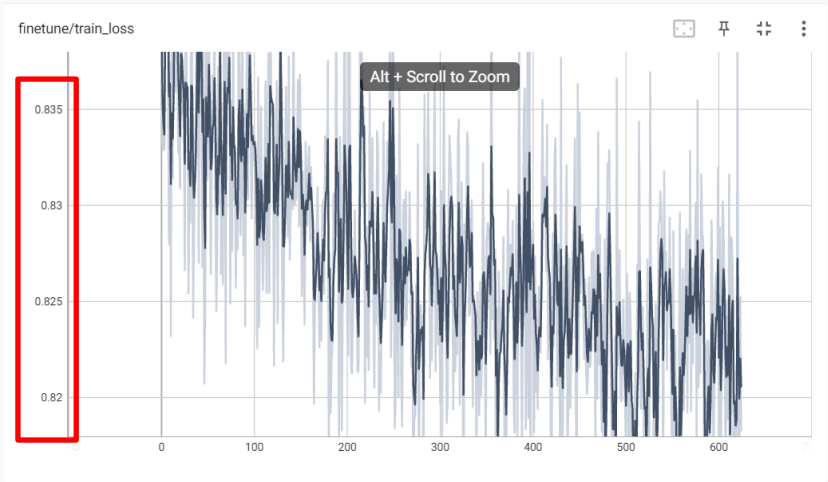

Top-5 Accuracy




Test loss




Train loss




```
Best top-1 acc: 94.73
```

# IV. Extra Works

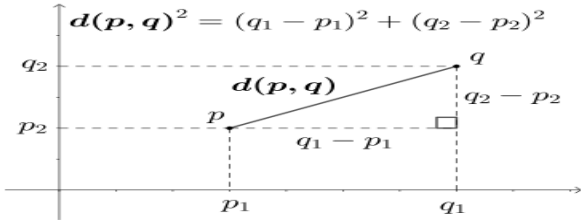
## EXTRA

### 코사인 유사도(Cosine Similarity) [12]

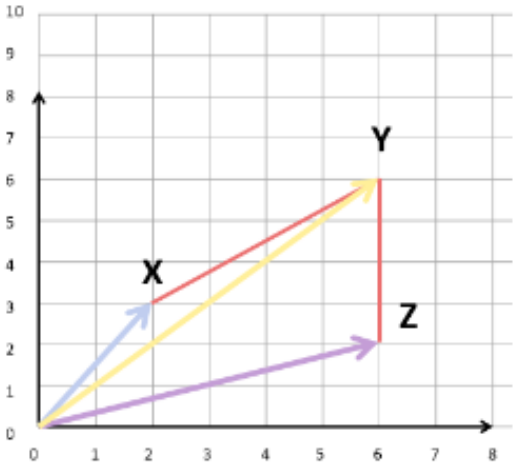

벡터의 크기에 영향을 받지 않고 벡터의 방향만을 고려
-> 데이터 간의 각도 차이를 기반


Y와의 유사도
X > Z

### 유클리디안 거리(Euclidean distance) [13]




두 특성 벡터 간의 L2거리사용(벡터의 크기)
-> 셀 값이 작은 차이에도 민감하게 반응




Y와의 유사도
X < Z

# IV. Extra Works

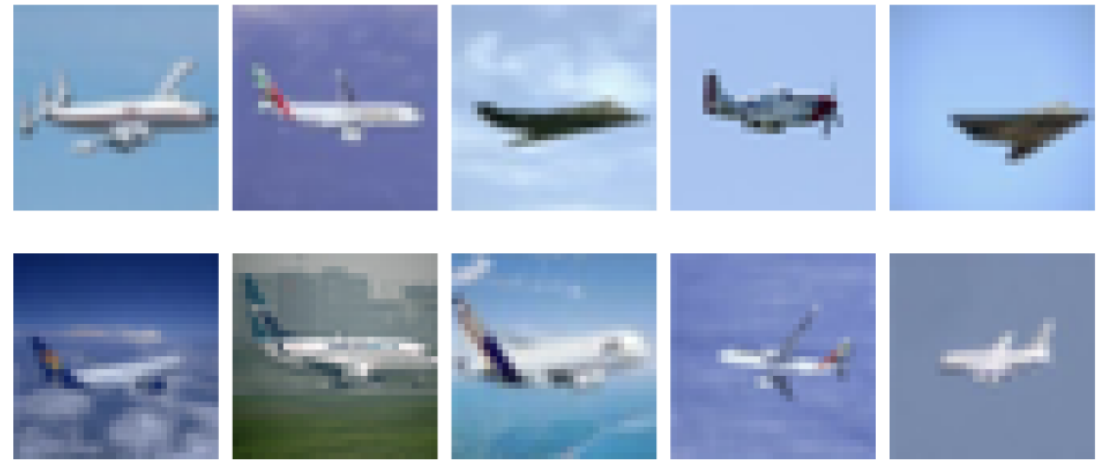
## EXTRA

```
image_path = '/content/drive/MyDrive/plane.PNG'

input_image = Image.open(image_path)
input_image
```


INPUT DATA

```
show_similar_images(similar_images)
```



학습 모델의 특징 기반
유사도 기반 CIFAR-10 DATA 추론

# V. References


[1] HE, Kaiming, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.

[2] YU, Fisher; KOLTUN, Vladlen; FUNKHOUSER, Thomas. Dilated residual networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 472-480.

[3] TAN, Mingxing; LE, Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. PMLR, 2019. p. 6105-6114.

[4] ZHOU, Bolei, et al. Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 2921-2929.

[5] KINGMA, Diederik P.; WELLING, Max. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[6] RADFORD, Alec; METZ, Luke; CHINTALA, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

[7] ALOYSIUS, Neena; GEETHA, M. A review on deep convolutional neural networks. In: 2017 international conference on communication and signal processing (ICCSP). IEEE, 2017. p. 0588-0592.

[8] THANH-TUNG, Hoang; TRAN, Truyen. Catastrophic forgetting and mode collapse in gans. In: 2020 international joint conference on neural networks (ijcnn). IEEE, 2020. p. 1-10.

[9] HO, Jonathan; JAIN, Ajay; ABBEEL, Pieter. Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 2020, 33: 6840-6851.

[10] HEUSEL, Martin, et al. Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems, 2017, 30.

[11] PHAM, Hieu, et al. Meta pseudo labels. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021. p. 11557-11568.

[12] SITIKHU, Pinky, et al. A comparison of semantic similarity methods for maximum human interpretability. In: 2019 artificial intelligence for transforming business and society (AITB). IEEE, 2019. p. 1-4.

[13] DOKMANIC, Ivan, et al. Euclidean distance matrices: essential theory, algorithms, and applications. IEEE Signal Processing Magazine, 2015, 32.6: 12-30.